Analysis of aerial images to protect marine ecosystems

“Detection and Classification”

ALIREZA FOROUTAN TORKAMAN

M1 Computer Science (Artificial intelligence)

Students ID: 22308001

Supervisor: Martinet Jean

2023/2024

Contents

Table of figures

Table of charts

ABSTRACT

This project presents a custom-trained YOLOv8 [1] (You Only Look Once) object detection model for detecting boats in satellite images, specifically annotated with data obtained from drone imagery captured by MARRES Master [2] students in the Lérins Islands [3] area. The model, implemented using the Ultralytics library, is trained on a dataset annotated with this drone imagery to enhance its accuracy in local maritime environments. Following training, the model is applied to new aerial images for boat detection, and the detected boat regions are processed to determine their real-world areas. The project integrates morphological [4] operations, visualization of results, and statistical analysis, including counts and categorization of different boat types. Users can manually set thresholds or utilize an automatic approach based on percentiles derived from the boat area distribution. This project serves as a specialized tool for boat detection and analysis in the Lérins Islands [3] area, contributing to applications in maritime surveillance and monitoring.

Figure 1 Lérins Islands area

INTRODUCTION

Welcome to a critical environmental initiative unfolding in the Iles de Lerins, where the picturesque setting faces a significant challenge – the influx of boats mooring in areas rich with Posidonia Oceanica [5], a protected Mediterranean species. Despite existing regulations and an eco-mooring park, the pressure on these vital habitats persists. Taking charge of this challenge are dedicated professionals from the Natura 2000 [6] marine protected area, supervised by the city of Antibes. Their mission: to strategize and implement measures that enhance mooring regulations, ensuring the preservation of Posidonia Oceanica.

At the core of this effort is the goal to ease mooring pressure on Posidonia Oceanica within the Marine Protected Area, with a particular focus on Lerins Island [3] due to its heightened vulnerability. The initial step involves establishing a clear baseline, encompassing current marine habitat coverage and understanding boat mooring patterns. This groundwork sets the stage for MARRES students [2] to propose solutions, including the conceptualization of a novel eco-mooring project.

Zooming in on boat detection and classification within the maritime environment, my focus narrows to the Lérins Islands. Employing advanced AI and machine learning techniques, I aim to overcome unique challenges in maritime surveillance.

1) State of Art

In our pursuit of advanced marine ecosystem protection, we leverage cutting-edge object detection techniques, notably YOLOv8 by Ultralytics [1]. This powerful tool excels in real-time object detection, specifically in identifying and classifying boats within the Posidonia meadows. Our dataset, enriched by aerial images from drones, satellites, and photographs taken by MARRES Master students, provides a comprehensive view of the Lerins Island [3] and its surroundings. The meticulous annotation process, utilizing tools like Label Studio [5], ensures precision in capturing distinctive features, contributing to a nuanced understanding of boat mooring patterns and their impact on the Posidonia meadows. This collaborative effort showcases the sophistication and accuracy required in addressing environmental challenges in this unique marine ecosystem.

2) Researches and model selection

The foundation of this project [5] lies in the selection and training of an appropriate object detection model for boat detection, with a focus on the Lérins Islands [3] region. Leveraging machine learning techniques, the specific requirements of maritime surveillance and the characteristics of the dataset were taken into account during the model selection process. YOLOv8 [1], a powerful machine learning model architecture, was chosen for its renowned real-time object detection capabilities. It has demonstrated promising results across various domains, including scenarios similar to maritime surveillance.

The decision to adopt YOLOv8 [1] was driven by its ability to efficiently detect objects in diverse environments and handle multiple classes simultaneously, making it particularly well-suited for situations involving various boat types. This adaptability ensures that the model can effectively learn and recognize patterns unique to the maritime environment, ultimately enhancing the accuracy and reliability of boat detection in the targeted region.

3) Data preparation

3.1. The source of database

The data for this project was gathered from aerial images to practically address the protection of Lérins Islands [3] marine area. To make the model highly relevant and accurate for the challenges in this locale, I intentionally chose images taken of this geographic region. These images, captured by MARRES Master [2] students in 2021 and 2023 using a drone, cover two key zones:

- Northeastern region of Saint Marguerite.
- Area between the islands.

In this dataset, we have around 150 images, each taken horizontally at a 90-degree angle. These images not only showcase detailed backgrounds but also feature boats of different sizes and types found in the marine environment. By training the model on this diverse dataset, it became adepted at recognizing and responding effectively to the specific characteristics of the local maritime landscape.

Figure 2 Image taken of the area by drone

3.2. Data annotation and augmentation

In refining the training data, I focused on the crucial task of data annotation, ensuring precision in marking bounding boxes around specific objects, particularly boats categorized into three groups: *moving boats*, *sailing boats*, and *motor boats*. The Label Studio [5] website served as a user-friendly platform for this annotation process, generating an annotated dataset accompanied by text files with bounding box coordinates in the YOLO format.

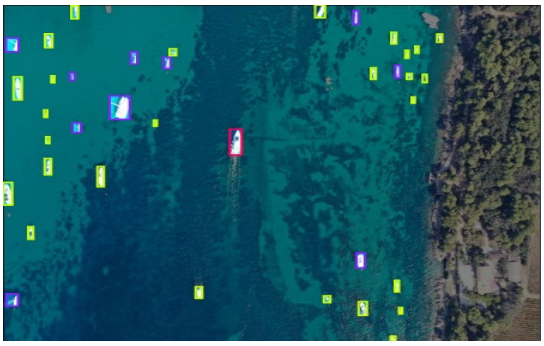

Figure 4 example of image annotation process

```
0 0.2236328125 0.495361328125 0.0185546875 0.0556640625
0 0.108642578125 0.354736328125 0.0263671875 0.046875
0 0.01806640625 0.187744140625 0.02294921875 0.0556640625
0 0.528076171875 0.077880859375 0.02587890625 0.0947265625
0 0.732421875 0.3115234375 0.0224609375 0.07763671875
1 0.816162109375 0.13525390625 0.0146484375 0.025390625
2 0.696044921875 0.755615234375 0.04541015625 0.076171875
0 0.328369140625 0.6650390625 0.0166015625 0.0703125
0 0.41748046875 0.904052734375 0.01123046875 0.048828125
0 0.31640625 0.845458984375 0.009765625 0.025390625
0 0.114013671875 0.5966796875 0.01513671875 0.03125
0 0.23876953125 0.8583984375 0.0078125 0.02734375
0 0.2587890625 0.928955078125 0.00732421875 0.0302734375
0 0.054931640625 0.708251953125 0.00830078125 0.02392578125
0 0.11083984375 0.73828125 0.01171875 0.0283203125
```

Figure 3 an example of .text file indicating boat locations

The images demonstrate how boats are labeled. The left picture shows a .text file indicating boat locations, with the first digit (0, 1, 2) representing the boat's class.

A substantial database for training models could significantly improve the accuracy and reliability of the model; To overcome the challenge of a limited initial dataset, I strategically employed data augmentation techniques. introducing the augmentor.py file, images underwent rotations of 180 degrees around the vertical and horizontal axis, effectively expanding the dataset from 150 to 357 images. Simultaneously, I resized images to strike a balance between model accuracy and training efficiency.

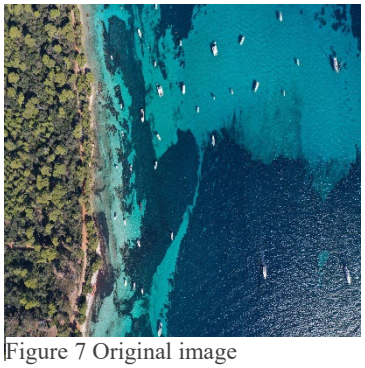

Figure 7 Original image

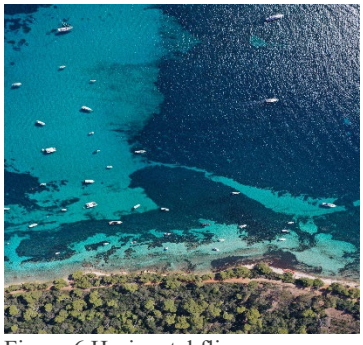

Figure 6 Horizontal flip

Figure 5 Vertical flip

4) Training and Evaluation Strategies

The program kicks off by running the main.py file, leveraging the Ultralytics library for machine training. Throughout the training process, I fine-tune crucial parameters such as epochs, image size, and batch number multiple times to achieve the best possible results. For validation, I used the 20% of database images specified in config.yaml.

```
model.train(data="config.yaml", epochs=300, imgsz=1280, batch=-1, device=[0])
```

```
      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
    300/300      2.28G     0.9766     0.6239     0.9272         33       1280: 100%|██████████| 109/109 [00:26<00:00,  4.07it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 55/55 [00:10<00:00,  5.08it/s]
                   all        434       3698      0.863      0.854      0.924       0.65

300 epochs completed in 10.580 hours.
Optimizer stripped from runs\detect\train2\weights\last.pt, 6.4MB
Optimizer stripped from runs\detect\train2\weights\best.pt, 6.4MB

Validating runs\detect\train2\weights\best.pt...
Ultralytics YOLOv8.0.196  Python-3.11.6 torch-2.1.1+cu121 CUDA:0 (NVIDIA GeForce RTX 3050 Ti Laptop GPU, 4096MiB)
YOLOv8n summary (fused): 168 layers, 3006233 parameters, 0 gradients, 8.1 GFLOPs
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 55/55 [00:10<00:00,  5.39it/s]
                   all        434       3698      0.873      0.857      0.928       0.66
                    MB        434       3099      0.932       0.93      0.975      0.733
                   MoB        434        241       0.82      0.755      0.864      0.531
                    SB        434        358      0.866      0.885      0.946      0.716
Speed: 1.1ms preprocess, 15.3ms inference, 0.0ms loss, 1.6ms postprocess per image
Results saved to runs\detect\train2
```

Figure 8 Model training information

The image above illustrates the conclusion of model training at the 300th epoch, highlighting the utilization of images sized at 1280 pixels

Following the training process, the main.py file outputs a file named "train" that contains a performance graph and the final trained models. These models include the best.pt (chosen based on the lowest error percentage), and the last.pt (representing the model trained in the final epoch).

After a thorough evaluation, where both models are tested on various images to assess their accuracy and reliability in detecting boats, it becomes evident that the last.pt model outperforms the best.pt model.

4.1. Evaluation metrics

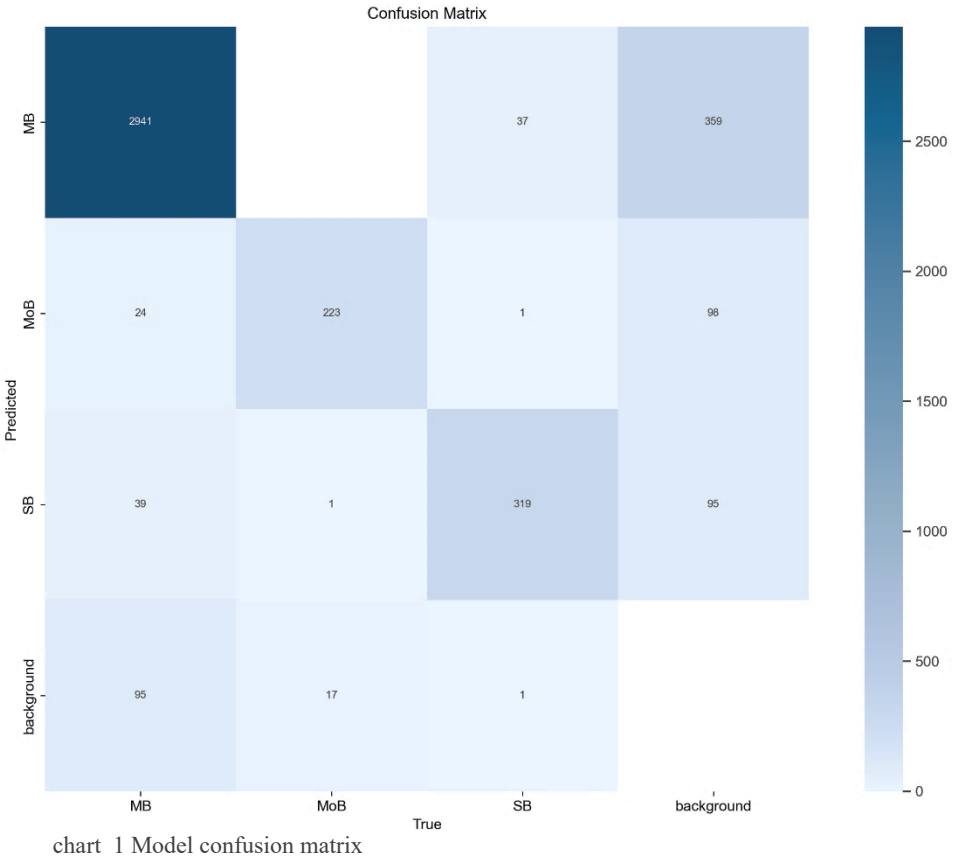

chart 1 Model confusion matrix

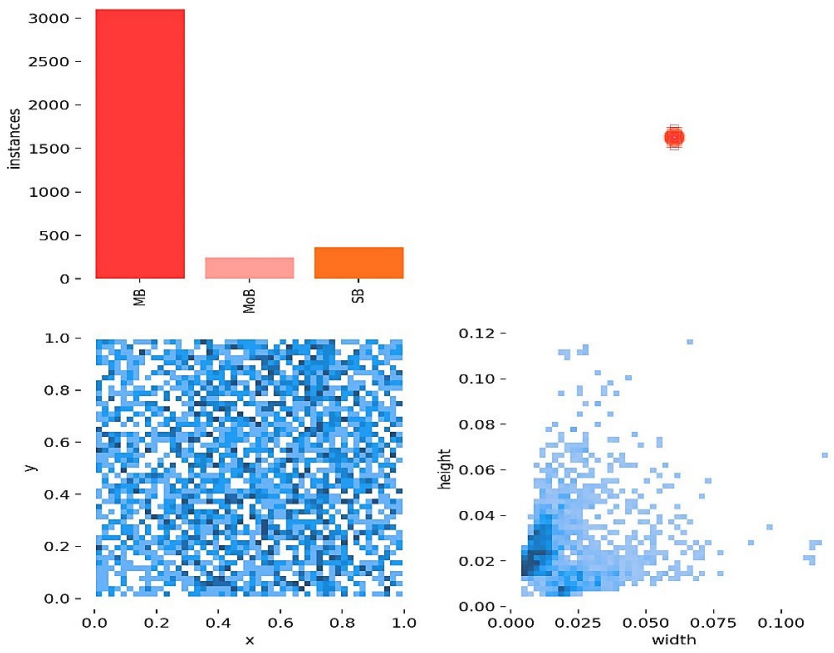

chart 2 labels

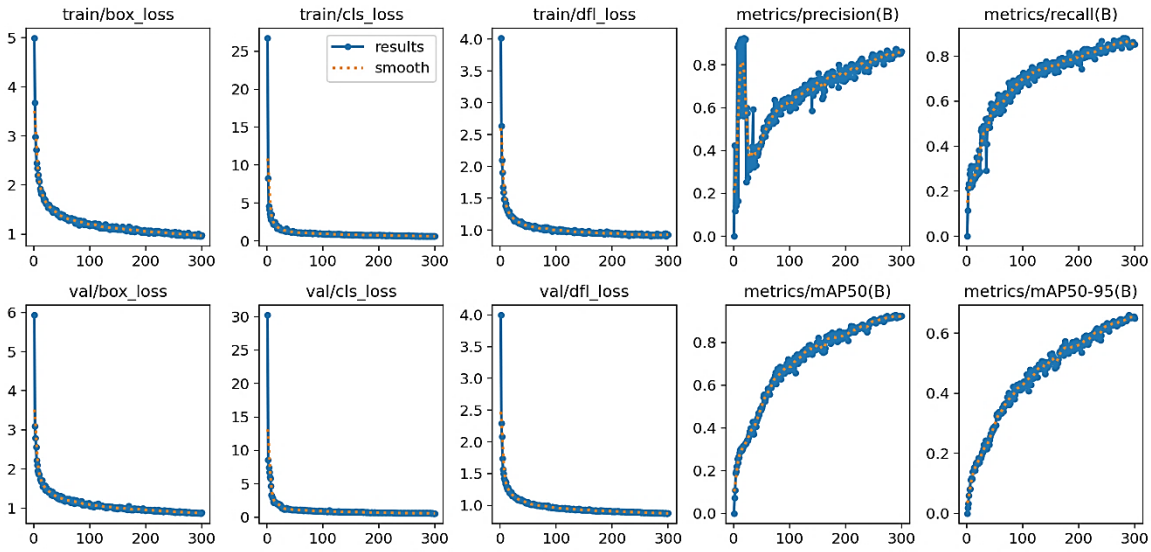

chart 3 Results

Examining the final performance metrics, we observe a remarkable reduction in the number of missing and undetected objects as the training progresses through epochs. This indicates that the model effectively learns and refines its ability to detect and identify objects from the custom dataset. Specifically, when evaluating the model using the matrix/map50(B) and matrix/map50/95(B) metrics, we note a consistent increase, signifying improved object detection performance at both moderate and strict IoU thresholds.

Further analysis of the validation metrics reveals noteworthy insights. The validation/dfl_loss, representing the object detection focal loss on the validation set, steadily decreases, indicating a reduction in the model's overall detection error. Concurrently, the validation/cls_loss and validation/box_loss metrics, pertaining to classification and bounding box regression losses, respectively, showcase a decline, affirming the model's proficiency in accurately classifying objects and precisely localizing their positions.

Meanwhile, scrutinizing the training metrics provides additional context. The train/dfl_loss, train/cls_loss, and train/box_loss exhibit a downward trend, showcasing the model's continuous improvement during the training process. This iterative enhancement emphasizes the adaptability of the model to the nuances and intricacies of the custom dataset. The decreasing values of these training losses further reinforce the notion that the model refines its object detection capabilities with each epoch.

5) Identification

After training the model, it is time to identify the boats in regional images, calculate their size, display the final result and their classification. All these steps are done by running the run.py file with the help of libraries for Object detection model, Image processing, Computer vision and Numerical operations. The process of identifying boats is done by simply placing the address of the image and the address of the model in the script and specifying parameters such as confidence percentage, result saving, etc.

```
image_path = 'c:/Users/lotus/Desktop/Zone 1 6pm 14.JPG'
model_path = 'runs/detect/train2/weights/last.pt'

# Load the model
model = YOLO(model_path)
results = model(image_path, conf=0.3, save=True, save_txt=True)
```

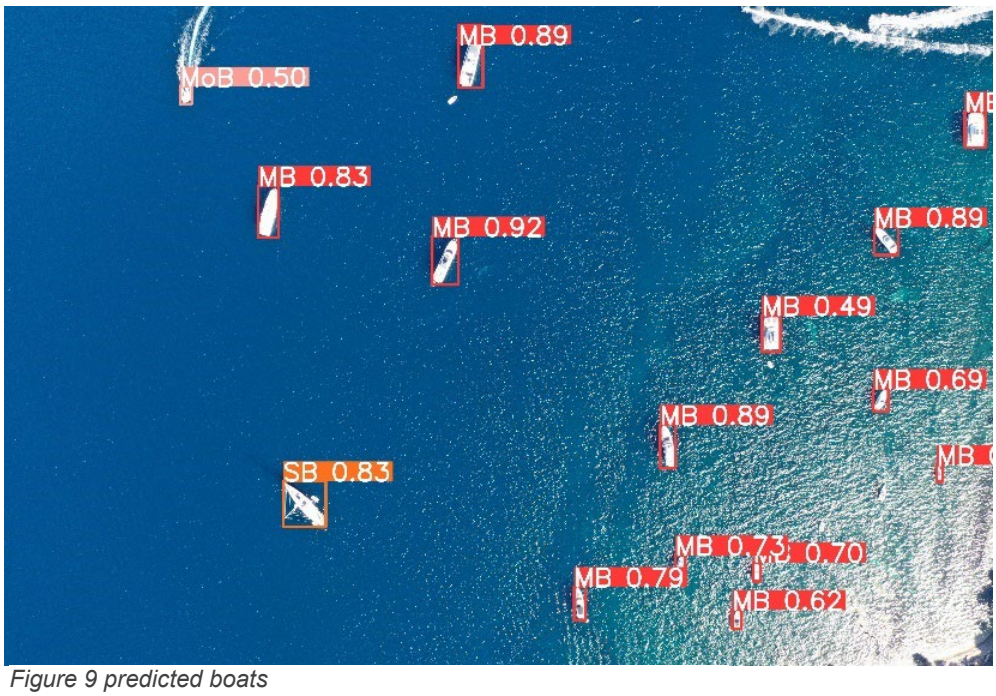

Figure 9 predicted boats

6) Area calculations:

6.1. Image metadata

First, to calculate the area of each boat, we need Loading and extracting metadata (imgsize, focal_length, altitude and sensor_size) from the input image. This is done automatically in the load_image function.

6.2. A Dual Approach

Due to the research nature of the project, I used two different approaches to calculate the area of the boats, and the user can choose any of these two methods to proceed. Both methods use YOLO coordinates which are in form of xyxy (x_min, y_min, x_max, y_max), where (x_min, y_min) are the coordinates of the top-left corner of the bounding box and (x_max, y_max) are the coordinates of the bottom-right corner.

6.3. First approach: Area_boxes()

This function is designed to determine the real-world area of a bounding box defined by YOLO coordinates through a multi-step process. It initiates by calculating the Ground Sample Distance (GSD) [9], a crucial metric representing the physical distance each pixel in the image corresponds to in the real world. This calculation involves the sensor size, altitude, and focal length. The GSD [9] values for both the x and y directions are then determined, providing a scale factor for converting pixel coordinates into real-world measurements.

```
# Calculate Ground Sample Distance (GSD)
gsd_x = (sensor_width * altitude) / (focal_length * imgsize[0])
gsd_y = (sensor_height * altitude) / (focal_length * imgsize[1])
```

This GSD-driven transformation is essential for accurately mapping image features to their corresponding physical dimensions.

Subsequently, the function utilizes these values to convert the pixel coordinates of the bounding box into real-world coordinates, facilitating a transition from image-based to real-world spatial representations.

Finally, the function computes the actual area of the bounding box in square meters, yielding a comprehensive understanding of the physical space occupied by the detected object.

6.4. Second approach: Area_pixels()

This function aims to estimate the real-world area of boats using a combination of YOLO coordinates and morphological operations [4].

6.4.1. Morphological Processing

The Morphology function serves as a crucial stage in the image processing pipeline, employing morphological operations [4] to enhance and refine the region of interest (ROI) extracted based on YOLO coordinates.
The process unfolds as follows:
Starting with the input image, the function isolates the ROI utilizing YOLO bounding box coordinates (x_min, y_min, x_max, y_max). This extracted ROI is then converted to grayscale for the sake of simplifying subsequent operations.
The process unfolds with the application of morphological operations [4], each serving a distinct purpose:
Opening, as the first operation, aids in smoothing the object boundaries and eliminating small protrusions.
Dilation, the subsequent operation, expands the object's features, contributing to the reinforcement of structural elements.
Erosion, the final morphological operation [4], shrinks the object boundaries, helping to eliminate noise and further refine the object contours.

Continuing the sequence, thresholding and masking steps are employed to create a binary mask that clearly isolates the object of interest against the background. Thresholding establishes a cutoff value for distinguishing between object and background pixels, while masking ensures that only the relevant portions of the image are retained.
In the concluding stages, the processed ROI (res) is obtained through the effect of these operations. This refined ROI serves as the foundation for precise pixel-based area calculations, contributing significantly to the overall accuracy and effectiveness of our image processing workflow.

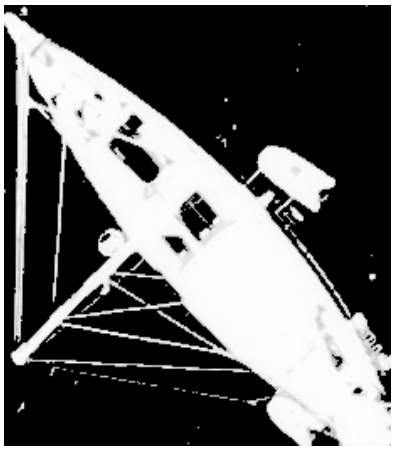

Figure 12 Morphological result on a sailing boat

Figure 11 Morphological result on a moving boat

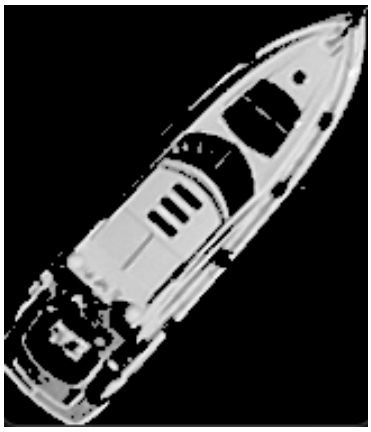

Figure 10 Morphological result on a motor boat

After image processing, The Area_pixels function smoothly integrates with the Morphology function by utilizing the processed Region of Interest (ROI). This involves calculating the object's area in terms of pixels, where we count the non-zero pixels within the processed ROI. Following this, we move on to computing Ground Sample Distances (GSD) based on sensor parameters and image size.
The next step in the process involves converting the pixel counts into real-world dimensions by applying scaling through the GSD values. This step ensures a seamless transition from pixel-based data to physical measurements.
Finally, to obtain the accurate real-world area of the detected object, we employ the geometric formula for area. This cohesive and continuous process ensures precision in determining the object's actual physical dimensions within the overall workflow.

6.5. Conclusion

Each method has its own set of advantages and drawbacks, and the choice between them depends on the specific characteristics of the dataset and images at hand.
In the first approach, determining the area of bounding boxes falls short of providing an accurate measure of the boats' total area, as it includes part of the sea surrounding the boats. Nevertheless, this method proves useful for comparing the relative sizes of the boats.

Conversely, the second method offers a more precise calculation of boat sizes. However, its accuracy heavily relies on the photo background, particularly in instances where sunlight reflection results in a matching color between the boats and the sea. This similarity in color could lead to inaccuracies in pixel count.
Additionally, computing the area of moving boats, marked by the white waves they create in the water, by counting non-zero and white pixels proves ineffective and fails to yield the actual boat area.

Given these limitations, in scenarios involving these complications, opting for bounding box area calculation is recommended over pixel-based methods.

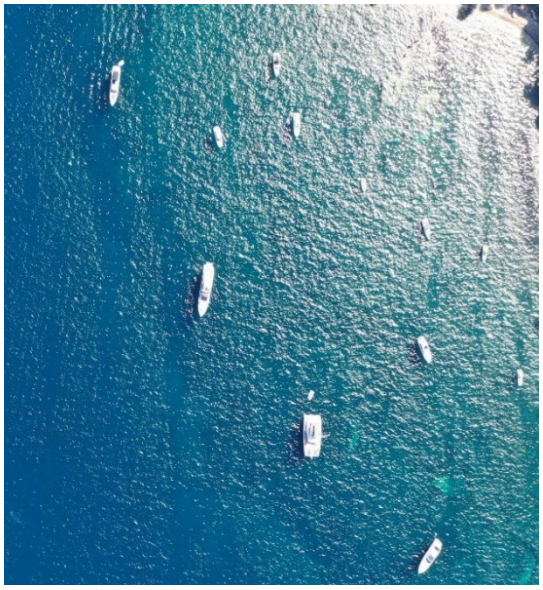

Figure 13 aerial image captured at 2:00 p.m.

Figure 14 shiny background

7) Visualization

This part is an important component of the project, focused on visualizing detected boats and their corresponding areas in an image. The visualization function begins by retrieving the results of object detection, including bounding box coordinates. It then enters a loop to iterate over each detected boat, printing the boat number and its associated area, derived from precomputed boat_areas. Within the loop, the code annotates the image with boat details, such as type, confidence percentage, number, area, and a visual representation of the boat's category using stars.

Additional information, including a legend for size categories and boat type abbreviations, is added to the image. The visualized image, stored as 'result_combined.jpg', is displayed and saved. Overall, this function enhances the interpretability of object detection results by providing a visually informative representation of detected boats and their characteristics.

Figure 15 Visualized sample

The displayed picture represents the ultimate result generated by the "visualization" function. Above each bounding box, information about the boat type and the confidence level of detection is presented. The confidence level, expressed as a percentage between 0 and 1, corresponds to the following boat categories:

- MoB: Moving Boat (motorboat / sailing boat)
- MB: Motor Boat (static)
- SB: Sailing Boat (static)

Beneath each bounding box, the computed area of the boat and the corresponding area ratio, denoted by '*', are displayed.

8) Data Analysis

The project provides a comprehensive analysis of detected boats in one or multiple images, including size distribution and type categorization. It allows users to choose between different methods for calculating boat areas and thresholds, providing flexibility for adapting to new datasets or dynamically adjusting parameters based on the current input. The visualization aspect enhances the interpretability of the results, aiding in a more detailed understanding of the distribution of boats in the analyzed images.

8.1. Results_and_Statistics.py

Results_and_Statistics.py is designed to conduct statistical analysis on an extensive set of images, specifically focusing on maritime images captured in the years 2021 and 2023 within the scope of this project. with the goal of providing valuable insights into boat detection, classification, and size distribution. The generated visualizations contribute to a comprehensive understanding of the maritime imagery dataset, laying the groundwork for further analysis and decision-making in marine-related applications.

The statistical analysis is divided into three main components:

- Distribution of categories:
  The script keeps track of counts for Moving Boats, Motor Boats, and Sailing Boats. These counts are visualized through bar charts, providing insights into the prevalence of each boat type in the dataset.
- Size Distribution:
  A histogram is generated to illustrate the distribution of boat sizes based on the calculated areas. This visualization helps understand the variability in boat sizes present in the dataset.
- Threshold Setting:
  The set_thresholds function automatically establishes class intervals and thresholds based on percentiles of boat sizes. These thresholds are utilized to categorize boats into different size classes for further analysis.

8.2. Setting thresholds

The 'set_thresholds' function is a pivotal component of the script, undertaking statistical analysis to automatically determine general class intervals and thresholds for boat size categorization. This process involves calculating crucial percentiles (25th, 50th, and 75th) of the boat areas, facilitating their division into distinct classes. Building on these percentiles, the function defines 'class_intervals,' outlining intervals for various classes, including [0, 25th percentile), [25th percentile, 50th percentile), [50th percentile, 75th percentile), and [75th percentile, +∞). The results of this analysis are then saved in 'class_intervals.txt,' a file that not only serves as documentation but is also utilized later in the 'plots' function within 'run.py' to set the class intervals for visual representations and analysis.

8.3. Statistics

The ‘plots’ function produces a comprehensive visualization of the project's outcomes. The first subplot displays the distribution of boat counts across size classes, offering an overview of the dataset's composition. The second subplot details the distribution of Moving Boats, Motor Boats, and Sailing Boats.
Additionally, a histogram depicts the distribution of boat sizes, aiding in identifying trends and patterns within the dataset.

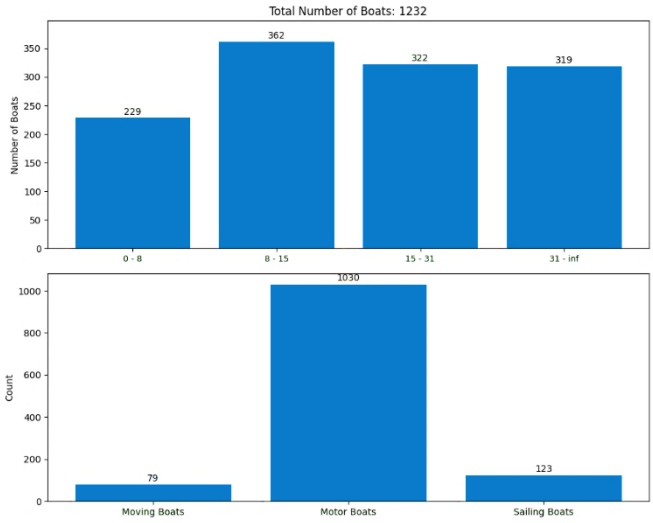

chart 5 statistic chart example

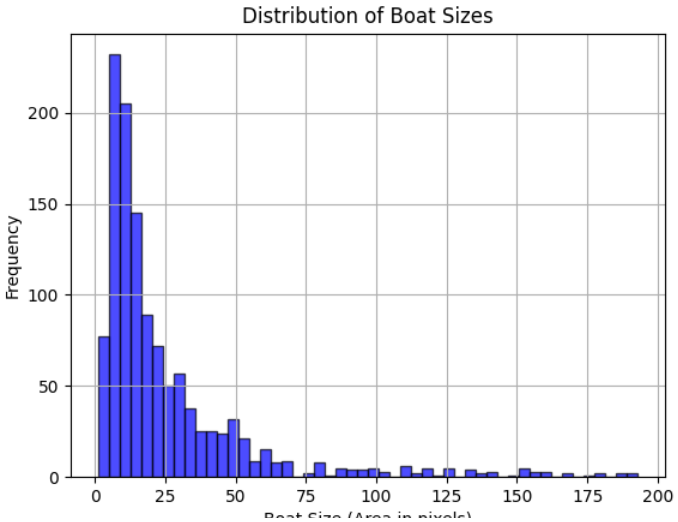

chart 4 distribution chart example

9) User interface:

The user interaction within the script revolves around making choices regarding the approach for calculating the area, as well as configuring classification thresholds and class intervals.
These choices include:

9.1. Area calculation

The user is prompted to choose between two methods for area calculation: Pixel base or Bounding box base.
1. If the user chooses pixels, the code calculates the area using Area_pixels().
2. If the user chooses boxes, the code calculates the area using Area_boxes().

9.2. Setting class intervals

9.2.1. Based on the collective area of all boats in the database

If the user selects this option, the program asks whether the user is working with a new dataset.
If it's a new dataset, the script imports Results_and_Statistics.py to utilize the General_thresholds function for reading class intervals from 'class_intervals.txt'.
If it's not a new dataset, the script continues without importing Results_and_Statistics.py and directly reads the class intervals from the 'class_intervals.txt' file.

9.2.2. Based on the range of boat areas present in the selected image

If the user selects this option, the script calls the Set_thresholds function.
The Set_thresholds function automatically determines class intervals and thresholds based on the distribution of boat areas calculated from the current input.
This option is suitable when the user wants to adapt the classification thresholds to the specific characteristics of the current image.

The efficient mechanism of saving and reusing classification thresholds in 'class_intervals.txt' streamlines the boat detection and classification process. This approach enhances the reproducibility and adaptability of the system, allowing users to seamlessly transition between datasets while maintaining consistency in size categorization. By providing a user-friendly choice to determine intervals or reuse existing ones, the script promotes flexibility in handling diverse maritime imagery, contributing to a more robust and adaptable analysis pipeline.

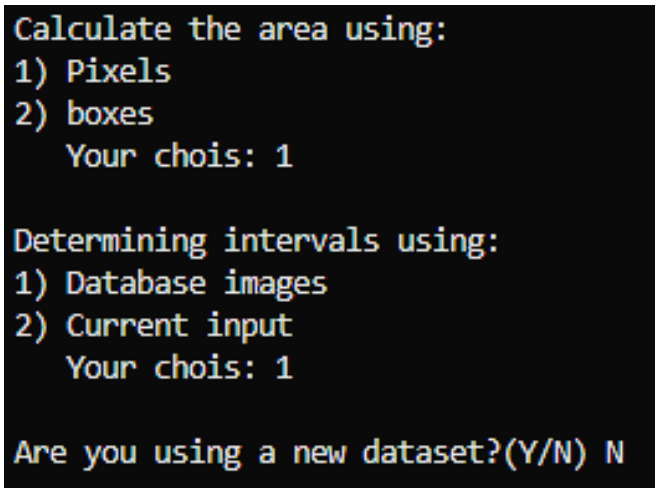

Figure 16 user interface

10) Model evaluation and progress

While executing the project, I aimed to achieve the best possible outcome through continuous adjustments to various parameters and extensive trial and error. In my latest endeavor, for instance, I enhanced the model by increasing the number of epochs from 300 to 400 and augmenting the size of the images.

A comparison of the results achieved with the current model as opposed to the previous one:

Figure 18 performance of the first model

Figure 17 performance of the second model

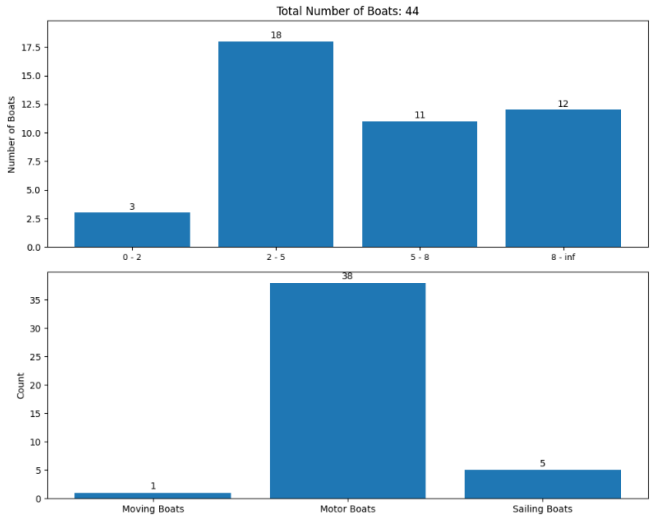

chart 7 performance of the first model

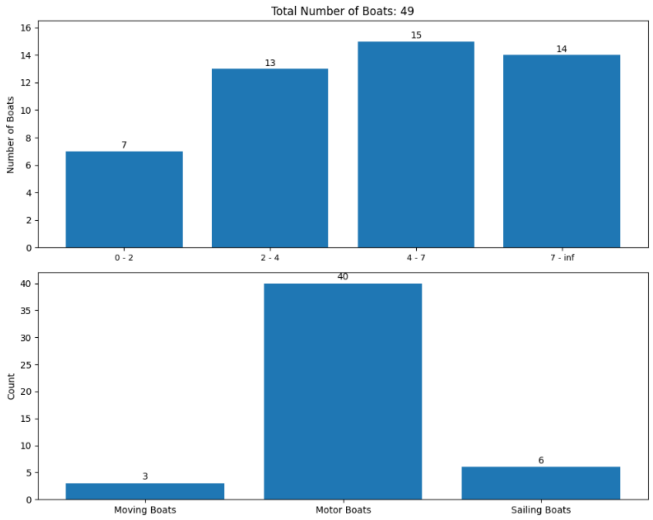

chart 6 performance of the second model

As evident from the results, the second model not only exhibits a notable improvement in the accuracy of detected boats but also demonstrates a higher count of identified boats compared to the first model.

11) Challenges

In the course of executing the project, I encountered multifaceted challenges that extended beyond technical intricacies, encompassing issues related to hardware limitations and the meticulous process of data annotation. Attempting to train the model on a local machine brought to light the delicate balance required between computational constraints and achieving optimal model performance.

To address hardware limitations, I engaged in extensive experimentation, seeking configurations that would reconcile the constraints of the local setup with the need for accurate model training. This iterative process not only honed my troubleshooting skills but also deepened my understanding of navigating challenges in resource-constrained environments.

The challenge of data labeling, a pivotal step in the training process, emerged as a time-consuming task demanding precision and meticulous attention. Labeling around 150 images, each potentially featuring multiple boats, underscored the critical role of comprehensive dataset preparation. This aspect not only demanded a substantial time investment but also underscored the importance of robust data preprocessing for the success of the entire machine learning pipeline.

Additionally, determining the real-world size of boats introduced another layer of complexity to the project. This task required the integration of diverse variables and mathematical equations to precisely calibrate the model's understanding of object dimensions in the physical world. Tackling this challenge deepened my awareness of the interdisciplinary nature of computer vision projects, where technical skills intertwine with domain-specific knowledge.

In summary, the project journey not only involved technical problem-solving but also spanned a broader range of skills, from optimizing hardware to meticulous data preparation and domain-specific considerations. The iterative process of overcoming these challenges significantly contributed to my ongoing learning and growth as a practitioner in the field of computer vision.

12) Methodology

The methodology involved careful research, selecting YOLOv8 as the object detection model, and gathering relevant aerial images for data preparation. Precise data annotation and augmentation enhanced the dataset, and machine training using the Ultralytics library fine-tuned the model. The identification process included boat identification, size calculation, and result visualization. In addition, to have a better understanding of the project and the area, we did on October 26th an expedition on Sainte-Marguerite island with Jean Martinet, Jean-François Maître (drone operator), and DS4H students of the project. We discovered the area, discussed the type of data we wanted, and helped the drone operator to find the best route for the drone to take the pictures. He did a successful test flight to acquire the first data with his drone, a DJI Mavic 3 Pro [10]. The report highlights result interpretability through visualization and statistical analysis, with continuous project evolution tracked through adjustments and comparative analysis. The successful implementation of the boat detection system is summarized, emphasizing ongoing improvements and recommendations for future enhancements, collaboration, and acknowledgments to contributors.

Figure 19 program performance on an aerial drone photo

13) Results And Discussion

Final statistics for all boats in the dataset revealed by Bounding Box Areas:

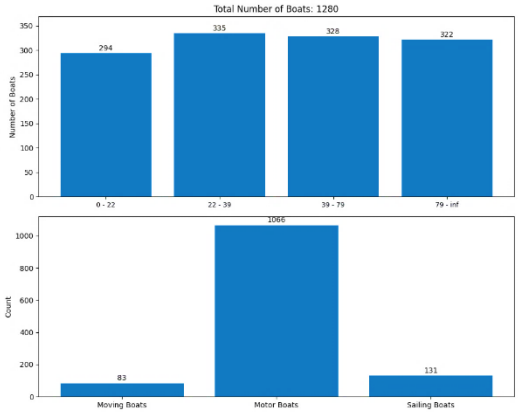

chart 8 class distributions (Bounding Box Areas)

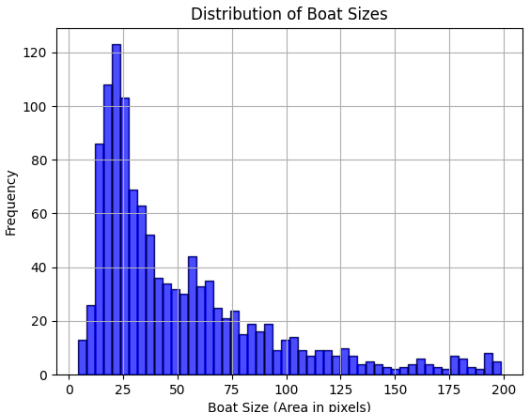

chart 9 size distributions (Bounding Box Areas)

Final statistics for all boats in the dataset revealed by counting pixels:

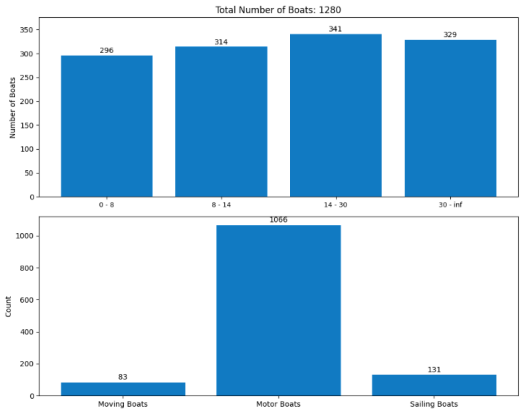

chart 11 class distributions (Pixel count)

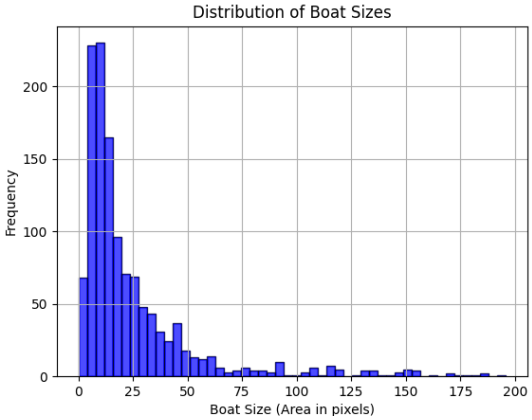

chart 10 size distributions (Pixel count)

The boat detection project in the Posidonia marine environment yielded significant outcomes, recognizing a total of 1,280 boats. Categorized into 83 in motion, 1,066 stationary motorboats, and 131 sailing boats, the project provided detailed insights into the boat distribution in the region.

Statistical analysis revealed different intervals for area calculation methods, with the second method showing shorter intervals, possibly indicating more accurate area measurements. This finding holds practical significance, especially for marine students, offering valuable insights into boat sizes in the Posidonia region.

14) Conclusion

In bringing the boat detection and classification project in the Posidonia marine environment to its conclusion, the project has delivered promising results with practical implications for both marine students and environmental surveillance. The detailed classification of boats, coupled with the recognition of motion statuses, provides a comprehensive overview of boat activity in the region. As the project moves forward, addressing identified limitations and incorporating feedback from MARRES [2] students and environmental experts will contribute to its continual refinement and application in real-world scenarios. Overall, the project represents a successful intersection of technology, education, and environmental monitoring, paving the way for informed decision-making and sustainable marine conservation efforts.

15) Acknowledgments

I would like to thank Jean Martinet for his supervision of us during this project, and Anne-Laure Simonelli for initiating the project and being our project coordinator. Special thanks to Jean-François Maitre for providing us with the aerial drone picture. My appreciation also goes to Christophe Mocquet and the students of the Master of Science, conservation, valorization of marine resources for providing us information about the Posidonia Oceanica. Finally, I wish to express my thanks to the other members of the Computer Science Master for the participation in the project.

16) References

[1] Ultralytics, "YOLOv8," 10 01 2023. [Online]. Available: docs.ultralytics.com/..

[2] U. o. c. d'azur, "MSC MARRES," [Online]. Available: https://univ-cotedazur.eu/msc/marres.

[3] M. d. l. v. d. cannes, "the lerins slands," [Online]. Available: https://www.cannes.com/en/boating-beaches/the-lerins-islands.html.

[4] OpenCV, "Morphological Transformations," [Online]. Available: docs.opencv.org/4.x/d4/d76/tutorial_js_morphological_ops.html..

[5] wikipedia, "Posidonia oceanica," [Online]. Available: en.wikipedia.org/wiki/Posidonia_oceanica.

[6] E. e. Agency, "The Natura 2000 protected areas network," [Online]. Available: www.eea.europa.eu/themes/biodiversity/natura-2000.

[7] I3S, "Analyse d'images aériennes pour la protection des écosystèmes marins," 2023. [Online]. Available: erebe, [Online]. Available: erebe-vm6.i3s.unice.fr:8080/eur-options/?id=533..

[8] Labelstud, "labelstud," [Online]. Available: labelstud.io/..

[9] PIX4D, "Ground sampling distance (GSD) in photogrammetry," [Online]. Available: support.pix4d.com/hc/en-us/articles/202559809-Ground-sampling-distance-GSD-in-photogrammetry.

[10] DJI, "DJI mavic 3 pro," [Online]. Available: www.dji.com/fr/mavic-3-pro.